Preconditioners and Tensor Product Solvers for Optimal Control Problems from Chemotaxis

Sergey Dolgov*and John W. Pearson†

June 22, 2018

Abstract

In this paper, we consider the fast numerical solution of an optimal control formulation of the Keller–Segel model for bacterial chemotaxis. Upon discretization, this problem requires the solution of huge-scale saddle point systems to guarantee accurate solutions. We consider the derivation of effective preconditioners for these matrix systems, which may be embedded within suitable iterative methods to accelerate their convergence. We also construct low-rank tensor-train techniques which enable us to present efficient and feasible algorithms for problems that are finely discretized in the space and time variables. Numerical results demonstrate that the number of preconditioned GMRES iterations depends mildly on the model parameters. Moreover, the low-rank solver makes the computing time and memory costs sublinear in the original problem size.

Keywords: *PDE-constrained optimization; Boundary control; Preconditioning; Chemotaxis; Mathematical biology*

1 Introduction

The process of chemotaxis in biology describes the movement of cells or organisms in a directed fashion as a response to external chemical signals. In 1971, Keller and Segel presented a mathematical model for bacterial chemotaxis [15]. In essence, for large numbers of bacteria, it is predicted that the bacteria will on average move up gradients of the chemoattractant concentration.

Since Keller and Segel's work, an area of numerical mathematics that has become a subject of significant interest is that of PDE-constrained optimization, where one wishes to predict the circumstances in which some physical (or in this case biological) objective occurs, subject to a system of PDEs describing the process. Using this technology, one is able to pose an inverse problem for the chemotaxis mechanism: given an observed bacterial cell concentration profile, what can be said about the external chemoattractant at the boundaries of a domain of interest? The constraints for this problem are therefore the PDEs describing bacterial chemotaxis. This is a parameter identification problem that has been considered in literature

*University of Bath, Claverton Down, BA2 7AY, Bath, United Kingdom. s.dolgov@bath.ac.uk

†School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom. j.pearson@ed.ac.uk

such as [19, 31], and in particular it was shown numerically by Lebiedz and Brandt-Pollmann that "it is possible to systematically control spatiotemporal dynamical behavior" [19].

The fast and efficient iterative solution of PDE-constrained optimization problems has increasingly become an active area of research, and in particular it is now widely recognised that the incorporation of effective preconditioners to accelerate iterative schemes is highly beneficial from a computational point-of-view. Preconditioning theory and numerics for a number of steady [27, 28, 29, 32, 37, 43] and time-dependent [3, 25, 26, 39] problems have been established, with [25, 39] describing the resulting solvers for reaction–diffusion problems from chemistry and biology. In this paper, we derive a potent preconditioner for the chemotaxis problem based on the saddle point structure of the matrix systems resulting from Newton-type iterations of the nonlinear PDEs.

When solving these optimization problems, which often involve the solution of a system of PDEs with initial conditions coupled with adjoint PDEs equipped with final-time conditions, there are many challenges arising from the time-dependent component of the problem in particular, due to the forward-backward solves required, and the associated scaling of computational complexity with the fineness of the grid in the time variable. Difficulties also arise from nonlinear problems, due to the matrices arising from the PDE system varying in structure at every time step, unlike linear problems for which some matrices can be re-used repeatedly within a solver. For time-dependent nonlinear problems that arise from chemotaxis, computer storage is therefore a significant bottleneck, unless a numerical algorithm is specifically tailored in order to mitigate this.

To combat this issue, in addition to presenting our new preconditioner, we describe an approach for approximating the solution of our problem in a low-rank format, namely the Tensor Train decomposition [22]. Low-rank tensor techniques emerge from the separation of variables and the Fourier method for solving PDEs. We can approximate the solution in the form $z(x, y) \approx \sum_{\alpha=1}^{r} v_\alpha(x) w_\alpha(y)$, using a possibly small number of terms r. In this case, the discretized univariate functions in the low-rank decomposition are much cheaper to store than the original multivariate function. The discretized separation of variables requires the low-rank approximation of matrices (for two variables x, y), or tensors (for three or more variables). Practical low-rank tensor algorithms employ robust tools of linear algebra, such as the singular value decomposition, to deliver an optimal low-rank approximation for a desired accuracy. Extensive reviews on the topic can be found in [9, 17].

The efficiency of low-rank decompositions depends crucially on the value of the rank r, which in turn reflects the structure of a function. Discontinuous functions, in particular level set functions, may require high ranks. However, smooth functions allow very accurate low-rank approximations, and hence a sublinear complexity of the inverse problem solution [36, 40]. The inverse problem implies driving the solution to a desired state, which usually has a simple (and hence low-rank) structure. Therefore, as long as we avoid discontinuous functions in our formulation, the low-rank techniques can be very efficient for the inverse problem. This is demonstrated in our computational experiments.

This paper is structured as follows. In Section 2 we describe the problem statement of which the numerical solution is considered. In Section 3 we present the structure of the matrix systems that result from the discretization of the system of PDEs. In Section 4 we present our preconditioning strategy for these systems, with numerical results provided in Section 5. We describe the low-rank tensor decomposition which is employed for the matrix systems in Section 6, with additional numerical experiments relating to this approach carried out in Section 7. Finally, concluding remarks are made in Section 8.

2 Problem statement

We examine the following problem describing the optimal control of a bacterial chemotaxis system, based on studies in literature such as [19] and [31, Chapter 13]:

$$\min_{z,c,u} \quad \frac{1}{2}\int_\Omega (z(\mathbf{x},T)-\widehat{z})^2 + \frac{\gamma_c}{2}\int_\Omega (c(\mathbf{x},T)-\widehat{c})^2 + \frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)} u^2 \tag{1}$$

subject to

$$\begin{aligned} \frac{\partial z}{\partial t} - D_z\nabla^2 z - \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right) &= 0 \qquad \text{on } \Omega\times(0,T), \\ \frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2} &= 0 \qquad \text{on } \Omega\times(0,T), \end{aligned}$$

equipped with the boundary conditions and initial conditions:

$$\begin{aligned} \frac{\partial z}{\partial n} &= 0 \qquad \text{on } \partial\Omega\times(0,T), \\ \frac{\partial c}{\partial n} + \beta c &= \beta u \qquad \text{on } \partial\Omega\times(0,T), \\ z(\mathbf{x},0) &= z_0(\mathbf{x}) \quad \text{on } \Omega, \\ c(\mathbf{x},0) &= c_0(\mathbf{x}) \quad \text{on } \Omega. \end{aligned}$$

This problem is solved on a space-time domain $\Omega\times(0,T)$ with boundary $\partial\Omega\times(0,T)$, and for $\Omega\subset\mathbb{R}^2$. The variables z, c denote *state variables*, corresponding to the bacterial cell density and chemoattractant concentration respectively, with u the *control variable*, $\widehat{z}$, $\widehat{c}$ given *desired states*, z_0, c_0 given initial conditions, and γ_c, γ_u, D_z, α, ρ, w, β given (positive) parameters. We highlight that, by construction of the problem, the control u in some sense relates to the gradient of chemoattractant concentration on the boundary of the domain of interest. The form of the boundary condition which enforces the control makes this a *boundary control problem*. In this PDE-constrained optimization model, we wish to discover what the profile of this control must be in order for the biological system to behave in a way prescribed by the desired states $\widehat{z}$, $\widehat{c}$.

Remark 1. Although derived similarly, the main difference between the works of [19] and [31] is that [19] considers solely the misfit between z and $\widehat{z}$, regularized by a term involving the final time T. We believe that the methods introduced in this paper are equally applicable to either cost functional.

We now consider the first and second derivatives of the Lagrangian[1]

$$\begin{aligned} \mathcal{L}(z,c,u,p,q) = \frac{1}{2}\int_\Omega (z(\mathbf{x},T)-\widehat{z})^2 + \frac{\gamma_c}{2}\int_\Omega (c(\mathbf{x},T)-\widehat{c})^2 + \frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)} u^2 \\ + \int_{\Omega\times(0,T)} p_\Omega\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z - \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\ + \int_{\Omega\times(0,T)} q_\Omega\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \\ + \int_{\partial\Omega\times(0,T)} p_{\partial\Omega}\left(\frac{\partial z}{\partial n}\right) + \int_{\partial\Omega\times(0,T)} q_{\partial\Omega}\left(\frac{\partial c}{\partial n} + \beta c - \beta u\right), \end{aligned}$$

[1]For ease of notation, we exclude initial conditions within the definition of the Lagrangian.

where p and q denote the adjoint variables corresponding to z and c, with p_Ω, q_Ω the components of p, q within the interior of Ω, and $p_{\partial\Omega}$, $q_{\partial\Omega}$ the components on the boundary. We arrive at the following system for the Newton formulations of the first-order optimality conditions:

$$\begin{aligned}
&\frac{\partial s_z}{\partial t} - D_z\nabla^2 s_z + \alpha\nabla\cdot\left(\nabla\left(\frac{1}{1+c}\right)s_z\right) + \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}s_c\right)z\right) \qquad (2)\\
&\qquad = -\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z - \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \quad \text{on } \Omega\times(0,T),\\
&\frac{\partial s_c}{\partial t} - \nabla^2 s_c + \rho s_c - 2w\frac{z}{(1+z^2)^2}s_z \qquad (3)\\
&\qquad = -\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \quad \text{on } \Omega\times(0,T),\\
&\gamma_u s_u - \beta s_q = -\left(\gamma_u u - \beta q\right) \quad \text{on } \partial\Omega\times(0,T), \qquad (4)\\
&\chi_{\Omega_T}(s_z) - 2wq\frac{1-3z^2}{(1+z^2)^3}s_z + \alpha\nabla\left(\frac{2c}{(1+c^2)^2}s_c\right)\cdot\nabla p \qquad (5)\\
&\qquad - \frac{\partial s_p}{\partial t} - D_z\nabla^2 s_p - \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla s_p - 2w\frac{z}{(1+z^2)^2}s_q\\
&\qquad = \widehat{z} - \left(\chi_{\Omega_T}(z) - \frac{\partial p}{\partial t} - D_z\nabla^2 p - \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right)\\
&\qquad\qquad \text{on } \Omega\times(0,T),\\
&-\alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)s_z\right) + \gamma_c\chi_{\Omega_T}(s_c) + \alpha p\nabla\cdot\left(\nabla\left(\frac{2c}{(1+c^2)^2}s_c\right)z\right) \qquad (6)\\
&\qquad - \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right)s_p - \frac{\partial s_q}{\partial t} - \nabla^2 s_q + \rho s_q\\
&\qquad = \gamma_c\widehat{c} - \left(\gamma_c\chi_{\Omega_T}(c) - \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right)\\
&\qquad\qquad \text{on } \Omega\times(0,T),
\end{aligned}$$

where s_z, s_c, s_u, s_p, s_q are the Newton updates for z, c, u, p, q, and $\chi_{\Omega_T}(\blacksquare)$ denotes a function that restricts the variable to time $t = T$. The boundary conditions for the state and adjoint variables are given by

$$\frac{\partial s_z}{\partial n} = 0, \quad \frac{\partial s_c}{\partial n} + \beta s_c = \beta s_u, \quad \frac{\partial s_p}{\partial n} = 0, \quad \frac{\partial s_q}{\partial n} + \beta s_q = 0 \quad \text{on } \partial\Omega\times(0,T),$$

with initial and final-time conditions

$$s_z(\mathbf{x},0) = 0, \quad s_c(\mathbf{x},0) = 0, \quad s_p(\mathbf{x},T) = 0, \quad s_q(\mathbf{x},T) = 0 \quad \text{on } \Omega,$$

assuming an initial guess is chosen that satisfies the initial conditions for z, c and the final-time conditions for p, q.

Remark 2. We highlight that there also exist chemotaxis problems which may be written in distributed control form. For example the work in [6], on the identification of chemotaxis models with volume-filling, considers (amongst others) a problem which may be interpreted

in our setting in the following way:

$$
\begin{aligned}
\min_{z,f} \quad & \frac{1}{2}\int_{\Omega\times(0,T)} (z-\widehat{z})^2 + \frac{\gamma}{2}\int_{\Omega\times(0,T)} \left[f^2 + |\nabla f|^2\right] \\
s.t. \quad & \frac{\partial z}{\partial t} - \nabla^2 z + f\nabla^2 c + \nabla f\cdot\nabla c = 0 \qquad on\ \Omega\times(0,T), \\
& -\nabla^2 c + c = z \qquad on\ \Omega\times(0,T), \\
& \frac{\partial z}{\partial n} - f\frac{\partial c}{\partial n} = 0 \qquad on\ \partial\Omega\times(0,T), \\
& \frac{\partial c}{\partial n} = 0 \qquad on\ \partial\Omega\times(0,T), \\
& z(\mathbf{x},0) = z_0(\mathbf{x}) \quad on\ \Omega,
\end{aligned}
$$

where γ is a positive constant, and $f(z)$ denotes the chemoattractant sensitivity. The challenge in this case is to discover the necessary profile of the function f in order to drive the chemoattractant to a particular state. We believe that variants of the techniques introduced in this paper could also be applied to this distributed control problem.

3 Matrix systems for Newton and Gauss–Newton

In this section, we describe the matrix systems which are obtained by discretization of the optimization problem (1) using the finite element method.

Concatenating the Newton equations (2)–(6), along with boundary conditions and initial/final-time conditions, gives a block matrix system of the following form:

$$
\begin{bmatrix}
\mathcal{L}_{zz} & \mathcal{L}_{zc} & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\
\mathcal{L}_{cz} & \mathcal{L}_{cc} & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\
0 & 0 & \gamma_u\cdot\mathrm{Id} & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\
\mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 & 0 \\
\mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0
\end{bmatrix}
\begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix}
\tag{7}
$$

$$
= \begin{bmatrix}
\widehat{z} - \left(\chi_{\Omega_T}(z) - \frac{\partial p}{\partial t} - D_z\nabla^2 p - \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right) \\
\gamma_c\widehat{c} - \left(\gamma_c\chi_{\Omega_T}(c) - \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right) \\
-\left(\gamma_u u - \beta q\right) \\
-\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z - \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\
-\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right)
\end{bmatrix},
$$

where

$$
\begin{bmatrix} \mathcal{L}_{zz} & \mathcal{L}_{zc} \\ \mathcal{L}_{cz} & \mathcal{L}_{cc} \end{bmatrix} = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) - 2wq\frac{1-3z^2}{(1+z^2)^3} & \alpha\nabla\left(\frac{2c}{(1+c^2)^2}\blacksquare\right)\cdot\nabla p \\ -\alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)\blacksquare\right) & \gamma_c\chi_{\Omega_T}(\blacksquare) + \alpha p\nabla\cdot\left(\nabla\left(\frac{2c}{(1+c^2)^2}\blacksquare\right)z\right) \end{bmatrix},
$$

$$
\begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial t} - D_z\nabla^2 + \alpha\nabla\cdot\left(\nabla\left(\frac{1}{1+c}\right)\blacksquare\right) & \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\blacksquare\right)z\right) \\ -2w\frac{z}{(1+z^2)^2} & \frac{\partial}{\partial t} - \nabla^2 + \rho\cdot\mathrm{Id} \end{bmatrix},
$$

$$
\begin{bmatrix} \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ \mathcal{L}_{cp} & \mathcal{L}_{cq} \end{bmatrix} = \begin{bmatrix} -\frac{\partial}{\partial t} - D_z\nabla^2 - \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla & -2w\frac{z}{(1+z^2)^2} \\ -\alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) & -\frac{\partial}{\partial t} - \nabla^2 + \rho\cdot\mathrm{Id} \end{bmatrix},
$$

with Id denoting the identity operator, and $\chi_{\partial\Omega}(\blacksquare)$ representing a function restricted to the boundary $\partial\Omega$.

As an alternative to solving the Newton system (7), it is possible to instead consider a Gauss–Newton approximation, where one neglects second derivatives within the $(1,1)$-block of the saddle point matrix as defined in Section 4. This results in the solution of systems

$$\begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\ \mathcal{L}_{pz} & \mathcal{L}_{qz} & 0 & 0 & 0 \\ \mathcal{L}_{pc} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0 \end{bmatrix} \begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix} = \mathbf{b}, \tag{8}$$

where $\mathbf{b}$ is the same right-hand side vector as in (7).

To be more explicit about the $\chi_{\Omega_T}(\blacksquare)$ and $\chi_{\partial\Omega}(\blacksquare)$ terms, the associated matrices contain entries of the form $\int_\Omega \phi_i \cdot \phi_j|_{t=T}$ and $\int_{\partial\Omega\times(0,T)} \phi_i \cdot \phi_j|_{\partial\Omega}$ respectively, for finite element basis functions $\{\phi_i\}$ of the same form for each PDE variable.

3.1 Additional control constraints

It is perfectly reasonable to add the following control constraint:

$$u_-(\mathbf{x},t) \le u \le u_+(\mathbf{x},t) \quad \text{a.e. on } \partial\Omega \times (0,T),$$

for given functions u_-, u_+, into the PDE-constrained optimization model. In other words, we prescribe that the chemoattractant must behave in a "sensible" (physical) way on the boundary of the domain of interest. One way in which we can tackle this additional term is to modify the cost functional (1) to add a Moreau–Yosida regularization term (see [12]) for the bound constraints, thereby minimizing instead

$$\begin{aligned} \min_{z,c,u} \quad & \frac{1}{2}\int_\Omega (z(\mathbf{x},T)-\widehat{z})^2 + \frac{\gamma_c}{2}\int_\Omega (c(\mathbf{x},T)-\widehat{c})^2 + \frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)} u^2 \\ & + \frac{1}{2\epsilon}\int_{\Omega\times(0,T)} |\max\{0,u-u_+\}|^2 + \frac{1}{2\epsilon}\int_{\Omega\times(0,T)} |\min\{0,u-u_-\}|^2, \end{aligned}$$

with ϵ a given (small) positive constant, chosen to enforce the control constraints efficiently.

When forming the Newton system in this setting, we will be required to solve systems relating to the finite element discretization of the following terms:

$$\begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} + \frac{1}{\epsilon}G_\Lambda & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\ \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0 \end{bmatrix} \begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix} = \widetilde{\mathbf{b}}, \tag{9}$$

where

$$\widetilde{\mathbf{b}} := \begin{bmatrix} \widehat{z} - \left(\chi_{\Omega_T}(z) - \frac{\partial p}{\partial t} - D_z\nabla^2 p - \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right) \\ \gamma_c\widehat{c} - \left(\gamma_c\chi_{\Omega_T}(c) - \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right) \\ \frac{1}{\epsilon}(G_{\Lambda_+}y_+ + G_{\Lambda_-}y_-) - (\gamma_u u - \beta q) \\ -\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z - \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\ -\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \end{bmatrix}.$$

Here, G_{Λ_+}, G_{Λ_-}, G_{Λ} denote projections onto the active sets $\Lambda_+ := \{i : u_i > (u_+)_i\}$, $\Lambda_- := \{i : u_i < (u_-)_i\}$, $\Lambda := \Lambda_+ \cup \Lambda_-$ (for the i-th node on the discrete level).

4 Preconditioning for Gauss–Newton matrix systems

In this section we focus on deriving effective preconditioners for the matrix systems (8) and (9) resulting from the Gauss–Newton method applied to the chemotaxis problem, both without and with additional control constraints.

We base our preconditioners on the well studied field of *saddle point systems*, which take the form [2]

$$\underbrace{\begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{10}$$

with A symmetric positive semidefinite in our case, and B having at least as many columns as rows. Two well-studied preconditioners for the system (10) are given by [11, 18, 21]

$$\mathcal{P}_D = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}, \qquad \mathcal{P}_T = \begin{bmatrix} A & 0 \\ B & -S \end{bmatrix},$$

where the (negative) *Schur complement* $S := BA^{-1}B^\top$. It is known [11, 18, 21] that, provided the preconditioned system is nonsingular, its eigenvalues are given by

$$\lambda(\mathcal{P}_D^{-1}\mathcal{A}) \in \left\{1, \frac{1}{2}(1 \pm \sqrt{5})\right\}, \qquad \lambda(\mathcal{P}_T^{-1}\mathcal{A}) \in \{1\},$$

with these results also holding for the block triangular preconditioner $\mathcal{P}_T$ even if A is not symmetric. Now, as $\mathcal{P}_D^{-1}\mathcal{A}$ is diagonalizable but $\mathcal{P}_T^{-1}\mathcal{A}$ is not, preconditioning with $\mathcal{P}_D$ ($\mathcal{P}_T$) yields convergence of a suitable Krylov subspace method in 3 (2) iterations, respectively.

In practice, however, $\mathcal{P}_D$ and $\mathcal{P}_T$ are not useful preconditioners, as the matrices A and S are computationally expensive to invert in general. We therefore instead seek preconditioners of the form

$$\widehat{\mathcal{P}}_D = \begin{bmatrix} \widehat{A} & 0 \\ 0 & \widehat{S} \end{bmatrix}, \qquad \widehat{\mathcal{P}}_T = \begin{bmatrix} \widehat{A} & 0 \\ B & -\widehat{S} \end{bmatrix},$$

where $\widehat{A}$ and $\widehat{S}$ denote suitably chosen approximations of the $(1,1)$-block A and Schur complement S. The objective here is that our Krylov method will not converge in 3 or 2 iterations, but just a few more, while at the same time ensuring that our preconditioner is much cheaper to invert. From this point, we focus our attention on preconditioners of block triangular form.

4.1 Construction of the preconditioner

We first examine the system (8), and place this in saddle point form (10) as follows:

$$A = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 \\ 0 & \gamma_c \chi_{\Omega_T}(\blacksquare) & 0 \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} \end{bmatrix}, \qquad B = \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) \end{bmatrix}.$$

Furthermore, let us decompose the blocks A and B into sub-blocks:

$$A = \begin{bmatrix} A_s & 0 \\ 0 & A_u \end{bmatrix}, \qquad B = \begin{bmatrix} B_s & B_u \end{bmatrix},$$

where

$$A_s = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 \\ 0 & \gamma_c \chi_{\Omega_T}(\blacksquare) \end{bmatrix}, \qquad B_s = \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} \end{bmatrix}, \qquad B_u = \begin{bmatrix} 0 \\ -\beta \chi_{\partial\Omega}(\blacksquare) \end{bmatrix}.$$

In this paper, A_u corresponds to the finite element discretization of the following operators:

$$A_u \leftarrow \begin{cases} \gamma_u \cdot \mathrm{Id} & \text{without control constraints,} \\ \gamma_u \cdot \mathrm{Id} + \frac{1}{\epsilon} G_\Lambda & \text{with control constraints,} \end{cases}$$

that is to say the block A_u is altered if we instead consider the matrix (9) incorporating control constraints. Note that, as the saddle point system is written, the matrix A is not invertible and the Schur complement S therefore does not exist. We hence consider a suitable re-ordering of the matrices (8) and (9) to enable us to utilize classical saddle point theory.

In particular, we observe that the matrix under consideration may be factorized as follows:

$$\begin{bmatrix} A_s & 0 & B_s^\top \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix} = \begin{bmatrix} I & -A_s B_s^{-1} B_u A_u^{-1} & A_s B_s^{-1} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \underbrace{\begin{bmatrix} 0 & 0 & S_\angle \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix}}_{\mathcal{P}},$$

where identity matrices I are of appropriate dimensions. Note that as Id corresponds to the identity operator on the continuous level, this will become a finite element mass matrix in the discrete setting. We then take $\mathcal{P}$ to be the foundation of our preconditioner. We define $S_\angle$ as the '*pivoted Schur complement*' [3, Section 3.3]

$$S_\angle = B_s^\top + A_s B_s^{-1} B_u A_u^{-1} B_u^\top. \tag{11}$$

We approximate this term within our preconditioner using the '*matching strategy*' devised in [26, 28, 29], which aims to capture both terms of the Schur complement within the preconditioner. The approximation reads as follows:

$$S_\angle \approx \widehat{S}_\angle := \Big(B_s^\top + \frac{1}{\eta} A_s\Big) B_s^{-1} \Big(B_s + \eta B_u A_u^{-1} B_u^\top\Big), \tag{12}$$

Note that the matrix product $B_s^\top B_s^{-1} B_s$ captures the first term $B_s^\top$ of $S_\angle$, and $\big(\frac{1}{\eta} A_s\big) B_s^{-1} \big(\eta B_u A_u^{-1} B_u^\top\big)$ matches exactly the second term $A_s B_s^{-1} B_u A_u^{-1} B_u^\top$. The positive constant η is chosen to 'balance' the first and last matrix factors, $B_s^\top + \frac{1}{\eta} A_s$ and $B_s + \eta B_u A_u^{-1} B_u^\top$, within the Schur complement approximation, so that the two terms in the remainder $S_\angle - \widehat{S}_\angle$ are approximately of the same norm. Two natural choices for this constant are

$$\eta = \sqrt{\frac{\|A_s\|}{\|B_u A_u^{-1} B_u^\top\|}} \quad \text{or} \quad \eta = \sqrt{\frac{\max(\mathrm{diag}(A_s))}{\max(\mathrm{diag}(B_u A_u^{-1} B_u^\top))}},$$

with the second such choice much cheaper to compute. Approximately solving for the matrix $B_s + \eta B_u A_u^{-1} B_u^\top$ is made tractable by the effective approximation of a mass matrix (or a mass

matrix plus a positive diagonal matrix) of the form A_u by its diagonal, see [24, Section 4.1] and [41].

Putting all the pieces together, we state our preconditioner

$$\widehat{\mathcal{P}} = \begin{bmatrix} 0 & 0 & \widehat{S}_{\angle} \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix},$$

incorporating the Schur complement approximation above. Due to the re-ordering of the saddle point system that we have undertaken, this is a suitable choice of preconditioner that captures the characteristics of the matrix under consideration.

4.2 Application of the preconditioner

Applying the inverse of the preconditioner, $\widehat{\mathcal{P}}^{-1}$, as is necessary within an iterative method, therefore requires three main operations:

1. Applying B_s^{-1}: This is equivalent to solving the forward problem, rather than the coupled optimization problem. In practice this is approached time-step by time-step, using an algebraic or geometric multigrid method, or another suitable scheme, to solve for the matrices arising at each point in time.

2. Applying A_u^{-1}: The matrix A_u is a block diagonal matrix, consisting of boundary mass matrices at each time-step (in the case without control constraints), or boundary mass matrices plus positive semidefinite diagonal matrices (if control constraints are present). In either case, these matrices may be well approximated using Chebyshev semi-iteration [7, 8, 42], or even using a simple diagonal approximation of a mass matrix [41].

3. Applying $\widehat{S}_{\angle}^{-1}$: Applying the approximation (12) involves a multiplication operation involving B_s, and (approximate) solves for each of $B_s^\top + \frac{1}{\eta}A_s$ and $B_s + \eta B_u A_u^{-1} B_u^\top$ which may again be approached at each time-step in turn using multigrid or another appropriate method.

4.3 Uzawa approximation

In practice, we make a further modification to the preconditioner $\widehat{\mathcal{P}}$ in order to ensure it is easier to work with on a computer. In more detail, the term B_s in the bottom-left of $\widehat{\mathcal{P}}$, and the terms $B_s^\top + \frac{1}{\eta}A_s$ and $B_s + \eta B_u A_u^{-1} B_u^\top$ within $\widehat{S}_{\angle}$, contain 2×2 block systems which we would like to replace with more convenient approximations so that we are only required to (approximately) invert one block at a time.

To facilitate this, we replace 2×2 block matrices by an inexact Uzawa approximation, with block triangular splitting matrices, where appropriate. This leads to our final choice of preconditioner:

$$\widehat{\mathcal{P}}_{\text{Uzawa}} = \begin{bmatrix} 0 & 0 & \left(B_s^\top + \frac{1}{\eta}A_s\right)_{\text{Uzawa}} (B_s)^{-1}_{\text{Uzawa}} \left(B_s + \eta B_u A_u^{-1} B_u^\top\right)_{\text{Uzawa}} \\ 0 & A_u & B_u^\top \\ (B_s)_{\text{Uzawa}} & B_u & 0 \end{bmatrix},$$

where $(\cdot)_{\text{Uzawa}}$ denotes the Uzawa approximation of the corresponding matrix. For ease of reproducibility for the reader, we state the splitting matrices below:

$$
\begin{aligned}
(B_s)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ 0 & \mathcal{L}_{qc} \end{bmatrix}, \\
\left(B_s^\top + \frac{1}{\eta} A_s\right)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{zp} + \frac{1}{\eta}\chi_{\Omega_T}(\blacksquare) & 0 \\ \mathcal{L}_{cp} & \mathcal{L}_{cq} + \frac{\gamma_c}{\eta}\chi_{\Omega_T}(\blacksquare) \end{bmatrix}, \\
\left(B_s + \eta B_u A_u^{-1} B_u^\top\right)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ 0 & \mathcal{L}_{qc} + \eta\beta^2 \chi_{\partial\Omega}(\blacksquare) A_u^{-1} \chi_{\partial\Omega}(\blacksquare)^\top \end{bmatrix}.
\end{aligned}
$$

Now the linear systems with diagonal blocks ($\mathcal{L}_{zp}$, $\mathcal{L}_{qc}$, and so on) can be solved directly. Note that it is also possible to annihilate another off-diagonal block instead within the Uzawa approximation. However, we have found that the approximations listed above yield fast convergence in the numerical experiments.

5 Numerical experiments with control constraints

In this section we benchmark the preconditioned Newton method. For our test problem, the initial distribution of bacterial cells is chosen as a sum of m_0 independent Gaussian peaks,

$$
z_0(x, y) = \sum_{i=1}^{m_0} \exp\left(-2560 \cdot \left[(x - x_i)^2 + (y - y_i)^2\right]\right), \tag{13}
$$

where the centers $\{x_i, y_i\}$ are chosen randomly on $[0, 1]^2$. The desired distribution at the final time $T = 1$ is linear,

$$
\widehat{z}(x, y) = \langle z_0 \rangle \cdot (x + y), \tag{14}
$$

normalized by the initial mass,

$$
\langle z_0 \rangle = \int_{[0,1]^2} z_0(x, y)\, \mathrm{d}x\mathrm{d}y,
$$

since the model conserves the normalization of z. Both initial and target concentrations c are zero. The experiments were run in MATLAB R2017b on one core of a 2.4GHz Intel Xeon E5-2640 CPU.

In this section, we set $m_0 = 50$ and the control constraints $u_- = 0$ and $u_+ = 0.2$, in accordance with [31]. The default regularization parameters are set to $\gamma_u = 10^{-3}$ and $\gamma_c = 0.5$. The stopping tolerance for the Newton iteration is set to 10^{-4}. Moreover, we decrease the Moreau–Yosida regularization parameter ϵ geometrically from 10^{-1} to 10^{-4} as the iteration converges. This gives more robust behavior of the Newton method.

The computational time is shown in Fig. 1 (left). We see that it grows cubically with respect to the uniform grid refinement, which is expected for a three-dimensional (2D space + time) problem. The number of Newton iterations is quite stable with respect to the grid size, ranging from 11 to 14 depending on a particular distribution of the random initial guess.

The transient control signal is shown in Fig. 1 (right). We notice that it is accurately confined within the prescribed constraints. However, this leads to a rather large misfit in the target cell density (Fig. 2). While the density follows the linear distribution $\widehat{z}$ correctly in the top right corner of the domain, in the left bottom corner we see an excessive density

Figure 1: Left: CPU time of the Newton solver for the constrained control. Right: $u(\mathbf{x}, T/2)$.

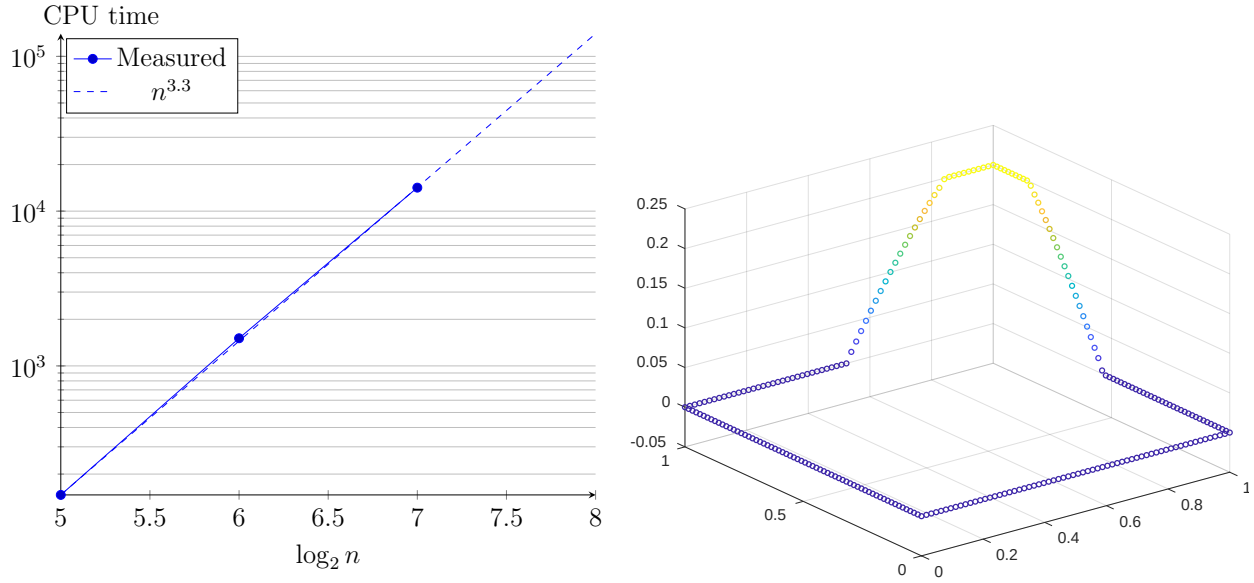
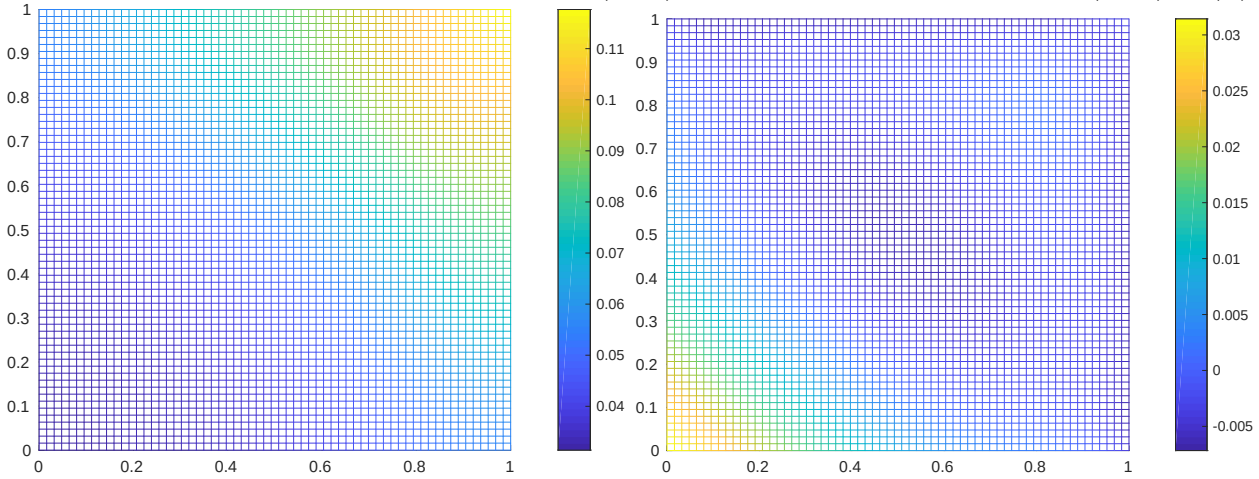


Figure 2: Left: cell density at the final time $z(\mathbf{x}, T)$. Right: misfit of the density, $z(\mathbf{x}, T) - \widehat{z}(\mathbf{x})$.



of bacteria. This shows that controlling only the chemoattractant might be insufficient for forcing the bacteria to leave a particular area.

Lastly in this section, we investigate the performance of the preconditioner proposed in Section 4 against variation of parameters. In Table 1, we show the average numbers of GMRES [33] iterations per Newton step for different grid sizes n and regularization parameters γ_u, γ_c. We vary only one parameter at a time, while the other two are kept fixed to their default values, $n = 64$, $\gamma_u = 10^{-3}$, and $\gamma_c = 0.5$. The number of iterations grows slightly as the control regularization parameter γ_u is decreased, which is expected for a boundary control problem. On the other hand, the preconditioner is reasonably robust with respect to the other parameters, in particular the grid size.

6 Low-rank tensor decompositions and algorithms

The optimality system (8) can result in a huge-scale matrix system, for many spatial degrees of freedom and time steps. One way to reduce the associated computational burden is to

Table 1: Average number of GMRES iterations per Newton step.

n	its
32	21.37
64	27.46
128	27.86

γ_u	its
10^0	6.00
10^{-1}	9.00
10^{-2}	15.28
10^{-3}	27.46
10^{-4}	46.84
10^{-5}	69.21

γ_c	its
0.5	27.46
10^{-1}	30.55
10^{-2}	32.11
10^{-3}	31.77
10^{-4}	32.67
10^{-5}	31.57

seek an approximate solution in a low-parametric representation. In this paper we apply separation of variables, and in particular the Tensor Train (TT) decomposition [22]. In this section, we introduce the TT decomposition and the algorithm for an efficient TT-structured solution of the optimality equations. Although the TT approximation can have difficulties with the indicator function of the active set of control constraints (see Remark 3 below), for the problem without box constraints it yields a very efficient solver. So in this section we assume an unconstrained control setting.

6.1 Tensor product discretization and indexing

We assume that the solution functions can be discretized on a structured grid, e.g. the cell concentration $z(\mathbf{x}, t)$ with a d-dimensional spatial variable $\mathbf{x} = (x_1, \dots, x_d)$ can be approximated by

$$z(\mathbf{x}, t) \approx \sum_{i_1,\dots,i_d,i_{d+1}=1}^{n_1,\dots,n_d,n_{d+1}} \mathbf{z}(i_1, \dots, i_d, i_{d+1}) \phi_{i_1,\dots,i_d}(\mathbf{x}) \psi_{i_{d+1}}(t),$$

where $\{\phi_{i_1,\dots,i_d}(\mathbf{x})\}$ is a set of spatial basis functions as introduced in Section 3, which we now assume to be indexed by d independent variables. In particular, we consider a square domain $\mathbf{x} \in [0, 1]^d$ and the piecewise polylinear basis functions

$$\phi_{i_1,\dots,i_d}(\mathbf{x}) = \varphi_{i_1}(x_1) \cdots \varphi_{i_d}(x_d).$$

In turn, $\{\psi_{i_{d+1}}(t)\}$ is a set of nodal interpolation functions in time, associated with the uniform time grid $\{t_{i_{d+1}}\}$, with $t_{i_{d+1}} = \tau \cdot i_{d+1}$, $i_{d+1} = 1, \dots, n_{d+1}$, and $\tau = T/n_{d+1}$. We can see that the discrete coefficients of z can be collected into a $(d+1)$-dimensional *tensor*. Introducing a uniform bound $n \geq n_k$, $k = 1, \dots, d+1$, we can immediately conclude that the tensor $\mathbf{z}$ has $\mathcal{O}(n^{d+1})$ entries. The computational complexity of solving (8) is usually much higher. This explains the sometimes relatively high computing times in the previous section.

Separation of the discrete variables $i_1, \dots, i_{d+1}$ can compress the tensor data from the exponential $\mathcal{O}(n^{d+1})$ to a linear volume $\mathcal{O}(dn)$. Yet we can aim for a higher compression ratio. Assuming that the range n_k of an index i_k is factorizable into a set of divisors $n_{k,1} \cdots n_{k,L_k} = n_k$, we can also factorize the index i_k into the corresponding digits,

$$i_k = 1 + \sum_{\ell=1}^{L_k} (i_{k,\ell} - 1) \prod_{p=1}^{\ell-1} n_{k,p}, \qquad k = 1, \dots, d+1.$$

Now the tensor $\mathbf{z}$ can be enumerated by the elementary digits $i_{k,\ell}$, which we shall denote simply as i_m from now on, for $m = 1, \dots, L = \sum_{k=1}^{d+1} L_k$. Instead of considering $\mathbf{z}$ as a $(d+1)$-dimensional tensor, we treat it as a L-dimensional tensor with elements $\mathbf{z}(i_1, \dots, i_L)$, and therefore we will separate now the *virtual* indices i_m [40].

6.2 Tensor Train decomposition

As a particular separated approximation, we choose the *Tensor Train* (TT) decomposition [22], which is also known as the Matrix Product States [30, 38] in physics:

$$\mathbf{z}(i_1, \dots, i_L) \approx \sum_{s_1=1}^{r_1} \cdots \sum_{s_{L-1}=1}^{r_{L-1}} z^{(1)}_{s_1}(i_1) z^{(2)}_{s_1,s_2}(i_2) \cdots z^{(L)}_{s_{L-1}}(i_L). \tag{15}$$

The factors $z^{(m)}$ on the right hand side are called *TT blocks*, and the ranges $r_1, \dots, r_{L-1}$ of the auxiliary summation indices are called *TT ranks*. Notice that the TT blocks are at most 3-dimensional tensors, of sizes $r_{m-1} \times n_m \times r_m$ (for uniformity, we can let $r_0 = r_L = 1$).

Potentially, we can represent any finite dimensional tensor exactly through (15) by choosing large enough TT ranks. For reasons of numerical efficiency, we will of course aim for a (sub-)optimal approximation with r_m being as small as possible, and most importantly much smaller than the original tensor size $n_1 \cdots n_{d+1}$. The storage needed for the right hand side of (15) is of the order of $\mathcal{O}(L n_m r_m^2)$, where $n_m = n_{k,\ell}$ is also chosen to be much smaller than the original n_k. For example, if we restrict the grid sizes to be powers of two, $n_k = 2^{L_k}$, the range of each index i_m in (15) becomes just $\{1, 2\}$, whereas L, and hence the storage complexity of the TT format, becomes *logarithmic* in the original tensor size, $L = \log_2(n_1 \cdots n_{d+1})$. Due to the minimal non-trivial index range in this case, the TT decomposition (15) with $i_m \in \{1, 2\}$ was called the *Quantized* TT (QTT) decomposition [16]. It was then proved that many examples of vectors [16] and matrices [13, 14], arising from the discretization of functions and differential operators, allow low-rank QTT decompositions.

Abstracting from the original problem dimensions, we can consider only two data representations: a tensor with the smallest possible ranges $\mathbf{z}(i_1, \dots, i_L)$, and a vector of the same data entries:

$$\mathbf{z}(i) = \mathbf{z}(i_1, \dots, i_L), \quad \text{where} \quad i = 1 + \sum_{m=1}^{L} (i_m - 1) \prod_{p=1}^{m-1} n_p. \tag{16}$$

We need the vector notation for setting the Gauss–Newton equations (8) on tensors consistently. Boldface letters (e.g. $\mathbf{z}$) from now on will denote vectors. We can use the Kronecker product ($\otimes$) to rewrite (15) in an equivalent vector form,

$$\mathbf{z} = \sum_{s_1, \dots, s_{L-1}=1}^{r_1, \dots, r_{L-1}} z^{(1)}_{s_1} \otimes z^{(2)}_{s_1,s_2} \otimes \cdots \otimes z^{(L)}_{s_{L-1}}.$$

Of course, we shall never actually compute the Kronecker products in the expansion above, but only store and manipulate individual TT blocks on the right hand side.

For example, the matrix-vector product $\mathbf{y} = A\mathbf{z}$ with $\mathbf{z}$ given in (15) can be computed efficiently if we can also represent the matrix by a TT decomposition,

$$A = \sum_{s_1, \dots, s_{L-1}=1}^{R_1, \dots, R_{L-1}} A^{(1)}_{s_1} \otimes A^{(2)}_{s_1,s_2} \otimes \cdots \otimes A^{(L)}_{s_{L-1}}. \tag{17}$$

For example if the matrix is diagonal, and the vector of the diagonal values can be represented by a TT decomposition (15), the matrix can be written as in (17), with the same TT ranks. There are less trivial matrices, arising for example in finite element computations, that admit TT decompositions with modest ranks R_m [13, 14]. Now the result $\mathbf{y} = A\mathbf{z}$ can be also written

in the TT format and computed block by block. Moreover, a TT decomposition with excessive TT ranks can be efficiently approximated up to a desired accuracy by a decomposition with sub-optimal ranks using QR and singular value decomposition (SVD) factorizations [22], without ever constructing full large tensors.

6.3 Alternating Linear Scheme iteration for solving (8)

In addition to the cell concentration $z(\mathbf{x}, t)$, we need to represent the other solution components. Since all components are defined on the same domain, we can discretize them using the same basis. The tensors of discrete values therefore have the same sizes. The structure of the problem (8) suggests that we approximate them in a shared TT decomposition, the so-called *block* TT format [5]. We denote the aggregated solution

$$\mathbf{y}^\top = \begin{bmatrix} \mathbf{z}^\top & \mathbf{c}^\top & \mathbf{p}^\top & \mathbf{q}^\top & \mathbf{u}^\top \end{bmatrix},$$

enumerating the components via $\mathbf{y}_j$, $j = 1, \ldots, 5$. Now we decompose $\mathbf{y}$ into a TT format with all the same TT blocks except the m-th block for some $m = 1, \ldots, L$, which actually carries the enumerator of the components,

$$\mathbf{y}_j = \sum_{s_1,\ldots,s_{L-1}=1}^{r_1,\ldots,r_{L-1}} y^{(1)}_{s_1} \otimes \cdots \otimes y^{(m-1)}_{s_{\ell-2},s_{\ell-1}} \otimes \widehat{y}^{(m)}_{s_{\ell-1},s_\ell}(j) \otimes y^{(m+1)}_{s_\ell,s_{\ell+1}} \otimes \cdots \otimes y^{(L)}_{s_{L-1}}. \tag{18}$$

Moreover, we can switch between the representations (18) corresponding to different m (and hence having j in different TT blocks) using the SVD [5]. For example, we can reshape $\widehat{y}^{(m)}$ into a matrix with elements

$$\widehat{Y}^{(m)}(s_{m-1}, i_m;\ j, s_m) = \widehat{y}^{(m)}_{s_{m-1},s_m}(i_m, j)$$

and compute the truncated SVD $\widehat{Y}^{(m)} \approx U\Sigma V^\top$. Now we write the left singular vectors U into the m-th TT block instead of $\widehat{y}^{(m)}$, and multiply $\Sigma V^\top$ with the $(m+1)$-th TT block,

$$y^{(m)}_{s_{m-1},s'_m}(i_m) = U(s_{m-1}, i_m;\ s'_m), \tag{19}$$

$$\widehat{y}^{(m+1)}_{s'_m,s_{m+1}}(i_{m+1}, j) = \sum_{s_m=1}^{r_m} \Sigma V^\top(s'_m;\ j, s_m) y^{(m+1)}_{s_m,s_{m+1}}(i_{m+1}). \tag{20}$$

Note that we have obtained the same representation as (18) with m replaced by $m+1$. This process can be continued further, or reversed, and hence the j-index can be placed into any TT block.

A crucial ingredient for the iterative computation of (18) is the *linearity* of the TT format. Having chosen an $m = 1, \ldots, L$, we construct the so-called *frame* matrix, where the TT block $\widehat{y}^{(m)}$ in (18) is replaced by the identity matrix,

$$Y_m = \sum_{s_1,\ldots,s_{m-2}} y^{(1)}_{s_1} \otimes \cdots \otimes y^{(m-1)}_{s_{m-2}} \otimes I_{n_m} \otimes \sum_{s_{m+1},\ldots,s_{L-1}} y^{(m+1)}_{s_{m+1}} \otimes \cdots \otimes y^{(L)}_{s_{L-1}}. \tag{21}$$

If we now treat $\widehat{y}^{(m)}(j)$ as a vector, we can observe that

$$\mathbf{y}_j = Y_m \widehat{y}^{(m)}(j), \tag{22}$$

i.e. the frame matrix realises a linear map from the elements of $\widehat{y}^{(m)}$ to the elements of the whole solution vectors. This motivates an iterative algorithm [10], which was called the Alternating Linear Scheme (ALS):

1: **for** iter $= 0, 1, \ldots$ until convergence **do**
2: **for** $m = 1, 2, \ldots, d, d-1, \ldots, 1$ **do**
3: Plug the solution in the form (22) into the original problem.
4: Solve the resulting overdetermined problem on $\widehat{y}^{(m)}$.
5: Prepare the format (18) and the frame matrix (21) for $m+1$ or $m-1$.
6: **end for**
7: **end for**

Starting from a low-rank initial guess of the form (18), this algorithm seeks the solution in a low-rank TT format by sweeping through the different TT blocks. However, there might be different ways to resolve the overdetermined problem in Line 4. For the optimality equations of the inverse problem, such as in (8), it was found [1, 4] to be efficient to use columns of the frame matrix as a Galerkin basis and project each submatrix of the Karush–Kuhn–Tucker (KKT) system individually. In our case we notice that the $(3,3)$-block of (8) is simply a diagonal matrix in the case where lumped mass matrices are considered, and therefore eliminate the control component from the equations.[2] Specifically, we deduce that $s_u = A_u^{-1}\left(\mathbf{b}_u + \beta\chi_{\partial\Omega}^\top s_q\right)$ and plug this into the fifth row. This gives us a system of 4 equations only. Moreover, instead of using the increments s_z, s_c, s_p, s_q, we can rewrite the equations for the new solution components directly:

$$\begin{bmatrix} \chi_{\Omega_T} & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c\chi_{\Omega_T} & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & 0 & -\beta^2\chi_{\partial\Omega}A_u^{-1}\chi_{\partial\Omega}^\top \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{c} \\ \mathbf{p} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{b}}_z \\ \widetilde{\mathbf{b}}_c \\ \widetilde{\mathbf{b}}_p \\ \widetilde{\mathbf{b}}_q \end{bmatrix},$$

where $\widetilde{\mathbf{b}}$ is the correspondingly adjusted right hand side. Now we plug in the solutions in the form (22) (with j now running only from 1 to 4), and project each of the previous equations onto Y_m. This gives us a *reduced* system

$$\begin{bmatrix} \widehat{\chi}_{\Omega_T} & 0 & \widehat{\mathcal{L}}_{zp} & \widehat{\mathcal{L}}_{zq} \\ 0 & \gamma_c\widehat{\chi}_{\Omega_T} & \widehat{\mathcal{L}}_{cp} & \widehat{\mathcal{L}}_{cq} \\ \widehat{\mathcal{L}}_{pz} & \widehat{\mathcal{L}}_{pc} & 0 & 0 \\ \widehat{\mathcal{L}}_{qz} & \widehat{\mathcal{L}}_{qc} & 0 & -\beta^2\widehat{\chi}_{\partial\Omega}^2 \end{bmatrix} \widehat{y}^{(m)} = \begin{bmatrix} Y_m^\top\widetilde{\mathbf{b}}_z \\ Y_m^\top\widetilde{\mathbf{b}}_c \\ Y_m^\top\widetilde{\mathbf{b}}_p \\ Y_m^\top\widetilde{\mathbf{b}}_q \end{bmatrix}, \tag{23}$$

with $\widehat{\chi}_{\Omega_T} = Y_m^\top\chi_{\Omega_T}Y_m$, $\widehat{\mathcal{L}}_{**} = Y_m^\top\mathcal{L}_{**}Y_m$ (where "$**$" stands for "zp", "zq", "cp", and so on), and $\widehat{\chi}_{\partial\Omega}^2 = Y_m^\top\chi_{\partial\Omega}A_u^{-1}\chi_{\partial\Omega}^\top Y_m$ the projected square matrices. Each submatrix is of size $r_{m-1}n_m r_m \times r_{m-1}n_m r_m$ (remember that we can choose $n_m = 2$), and hence (23) is easy to solve. Moreover, the singular value decomposition in (19)–(20) maintains the orthogonality of the frame matrices Y_m automatically in the course of alternating iterations, provided that the initial guess is given with this property. This makes the projected submatrices well conditioned if the original matrices were so, which eventually makes the entire matrix in (23) invertible. We highlight that the preconditioner developed in Section 4 can also be used for solving the system (23).

[2] Our derivation is of course valid for any invertible matrix A_u, however we wish to exploit the simplicity of the matrix structure within our solver. When consistent mass matrices are applied, we can well approximate these by their diagonals within a preconditioner, see [41].

6.4 Construction of matrices in the TT format

In the course of the Newton iteration, we need to reconstruct the matrices in (8) (and consequently in (23)) using the new solution. Assume that we need to construct an abstract bilinear form of a nonlinear transformation f of the solution,

$$\mathcal{L}_{\mathcal{B}} = \int f(z, c, p, q) \nabla^p \phi_i \cdot \nabla^q \phi_j \, \mathrm{d}\mathbf{x}, \tag{24}$$

where $p, q \in \{0, 1\}$ are the differentiation orders, and ϕ_i, ϕ_j are the basis functions. Instead of the exact functions z, c, p, q, we work with the tensors of their values, $\mathbf{z}, \mathbf{c}, \mathbf{p}, \mathbf{q}$. The corresponding values of f can be also collected into a tensor $\mathbf{f}$ of the same size, and the original function can be approximated in the same basis, i.e.

$$f(z(\mathbf{x}), c(\mathbf{x}), p(\mathbf{x}), q(\mathbf{x})) \approx \sum_{i_1, \ldots, i_d} \mathbf{f}(i_1, \ldots, i_d) \phi_{i_1, \ldots, i_d}(\mathbf{x}).$$

Now the computation of (24) involves computing analytical triple products

$$\mathcal{H}(i, j, k) = \int \phi_k \nabla^p \phi_i \cdot \nabla^q \phi_j d\mathbf{x}, \qquad i, j, k = 1, \ldots, (n_1 \cdots n_d), \tag{25}$$

and summing them up with the values of $\mathbf{f}$,

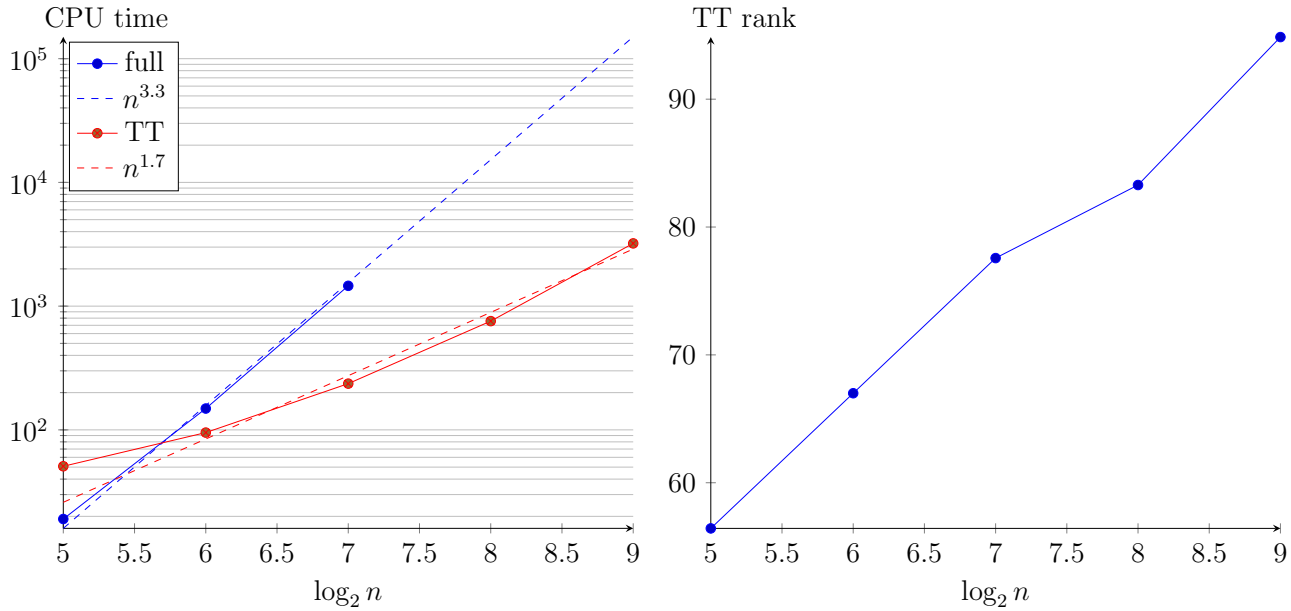
$$\mathcal{L}_{\mathcal{B}}(i, j) = \sum_{k=1}^{n_1 \cdots n_d} \mathcal{H}(i, j, k) \mathbf{f}(k). \tag{26}$$

Notice that we assume the basis functions can be enumerated by d independent indices, i.e. i is equivalent to $(i_1, \ldots, i_L)$ through (16), and similarly for j and k. The triple elements (25) therefore admit a TT decomposition (or even a single Kronecker-product term) similar to (17). Now, if the $\mathbf{f}$ tensor can also be approximated in the TT format (15), the bilinear form (24) can be represented in this format, with the TT ranks proportional (or equal) to those of $\mathbf{f}$. Moreover, the sum in (26) factorizes into individual sums over $k_1, \ldots, k_L$, which can be implemented efficiently block by block.

It remains to compute a TT approximation of $\mathbf{f}$. From the previous Newton iteration we are given the TT representation (18) for $\mathbf{z}, \mathbf{c}, \mathbf{p}, \mathbf{q}$. Hence we can rapidly evaluate any element of the solution components, and afterwards the corresponding value of f. In order to construct a TT approximation to $\mathbf{f}$ using only a few evaluations of f, we use the TT-Cross algorithm [23]. This is similar to the Alternating Linear Scheme outlined above, except that at each step it draws $r_{m-1} r_m$ fibers of the tensor values in the m-th direction in order to populate the m-th TT block and prepare the optimized fibers for the next step. In total it evaluates $\mathcal{O}(Lr^2)$ elements of the tensor, which is feasible under our assumption of small TT ranks. More robust and rank adaptive generalizations of this algorithm have followed [20, 34, 35].

Remark 3. Forming the diagonal of the indicator matrix G_λ in (9) seems also to be a task for the TT-Cross algorithm. However, it is likely to perform poorly in this setting, for two reasons. Firstly, if the discontinuity in a function, e.g. $\max\{0, u - u_+\}$, is not aligned to coordinate axes, the corresponding TT approximation requires very large TT ranks. This can be seen already in a two-dimensional case: a triangular matrix with all ones in one of the triangles is full-rank. Secondly, the TT-Cross algorithm is likely to overlook the part of the

Figure 3: CPU time (sec.) (left) and TT ranks (right) for different grid sizes n, $m_0 = 3$ initial peaks, accuracy threshold $\varepsilon = 10^{-4}$.



active set which is not covered by the initial (e.g. random) set of samples. In order to adapt the sampling fibers, the cross methods require a low discrepancy between adjacent tensor elements, which is not the case for G_λ. For this reason, we apply the TT approach only to the case of the unconstrained control.

7 Numerical experiments with the low-rank approximations

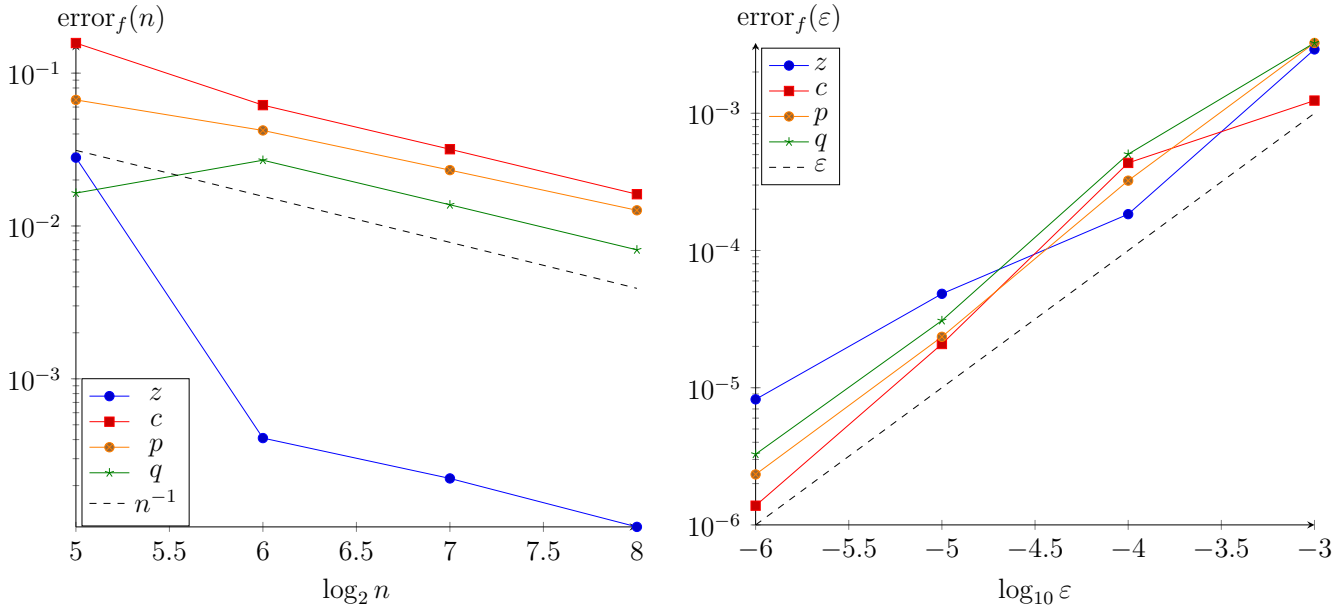
In this section, we benchmark the TT algorithm and compare it to the solver with the full vector representation. The initial distribution of bacterial cells z_0 and the desired state $\widehat{z}$ are chosen as in (13) and (14), with initial and target concentrations for the chemoattractant c set to zero. In this section the model is solved with an unconstrained control u, and final time $T = 1$. For the TT computations we used the TT-Toolbox implementation (see https://github.com/oseledets/TT-Toolbox).

7.1 Benchmarking of full and low-rank solvers

First, we compare CPU times of the original scheme that stores full vectors with those of the approximate TT solver (see Fig. 3). We fix $m_0 = 3$ randomly positioned Gaussian peaks in the initial distribution z_0. Since the particular ranks and numbers of iterations depend on the choice of z_0, we average the results over 8 realizations of z_0, for each value of n.

The cost of the full-format solver grows slightly faster than cubically, which is expected for a three-dimensional problem. This concerns both the CPU time and the memory. In particular, we could not run the full solver for $n > 128$ due to the memory limitations. On the other hand, the TT solver can proceed to much finer grids with lower time and memory footprint.

Figure 4: Left: discretization errors for different grid sizes n and $\varepsilon = 10^{-6}$. Right: approximation errors for different thresholds ε and $n = 64$.



7.2 Discretization and TT approximation errors

In order to justify the use of very fine grids (up to $n = 512$), let us estimate the discretization errors. In Fig. 4 (left), we vary the grid levels and plot the relative difference of the solutions on the grids with n and $2n$ points in each direction,

$$\text{error}_f(n) = \frac{|\langle f_n^2 \rangle - \langle f_{2n}^2 \rangle|}{\langle f_{2n}^2 \rangle}, \qquad \langle f_n^2 \rangle = f_n(T)^\top M f_n(T),$$

where $f_n(T)$ is the final-time snapshot of the solution component $f \in \{z, c, p, q\}$, computed at the grid with n nodes in each variable, and M is the mass matrix in space. The number of initial peaks $m_0 = 3$ and their positions are fixed in these experiments.

We see that the error decays linearly with respect to n, as expected from the implicit Euler scheme, for all quantities. Since this decay is rather slow, at least 256 points in each direction are necessary to achieve an accuracy of 1% in q, and hence the control, u.

The truncated singular value decomposition in the TT algorithm tries to introduce the same average amount of error to all solution components. However, the relative error in each component may differ from ε, depending on the norm scale and other factors of the algorithm, such as the local system solver. In Fig. 4 (right) we investigate the relative error in all components $f \in \{z, c, p, q\}$,

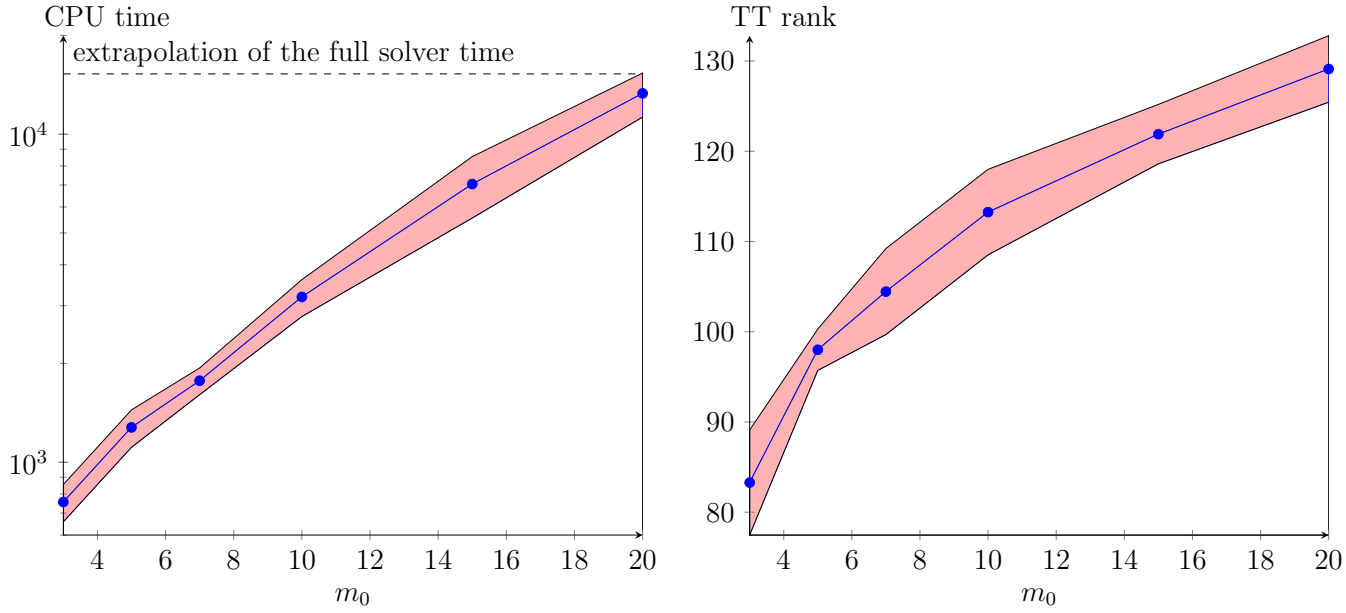
$$\text{error}_f(\varepsilon) = \frac{\|f_\varepsilon - f_{10^{-8}}\|_F}{\|f_{10^{-8}}\|_F},$$

where f_ε is the solution vector computed with the TT approximation threshold ε. We see that, on average, the errors decay linearly with ε, as expected.

7.3 Number of peaks in the initial distribution

Since the initial distribution of cells (13) consists of several randomly located Gaussian peaks, the particular positions of the peaks may influence the performance of the methods. In Fig.

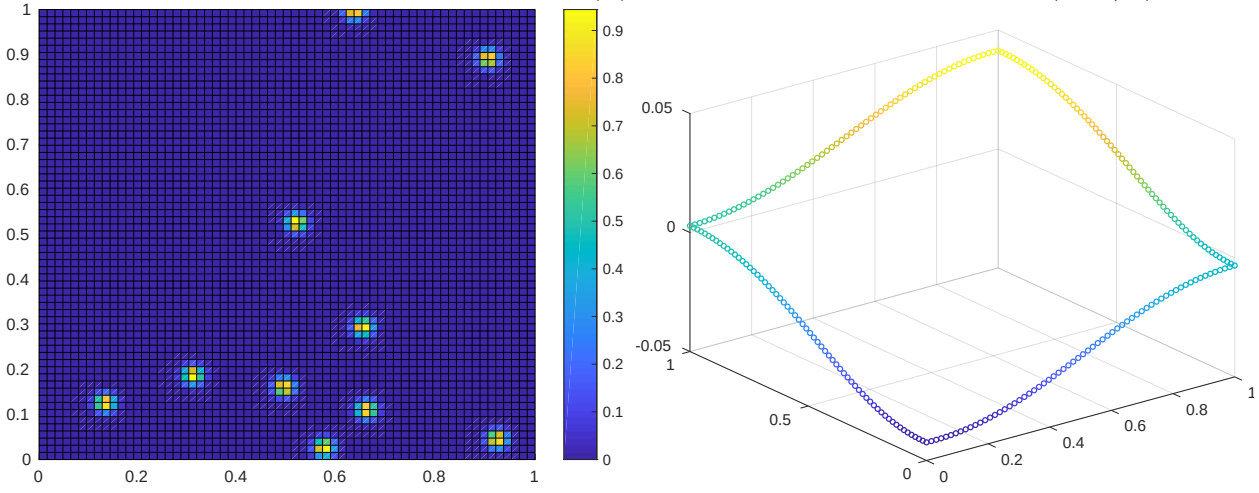
Figure 5: CPU time (sec.) (left) and TT ranks (right) for different numbers m_0 of initial peaks, accuracy threshold $\varepsilon = 10^{-4}$, $n = 256$.



5 we investigate CPU times and TT ranks in the TT solver versus the number of peaks m_0 and their positions. The plots show means plus minus standard deviations of the times and ranks with respect to the randomization of peak locations.

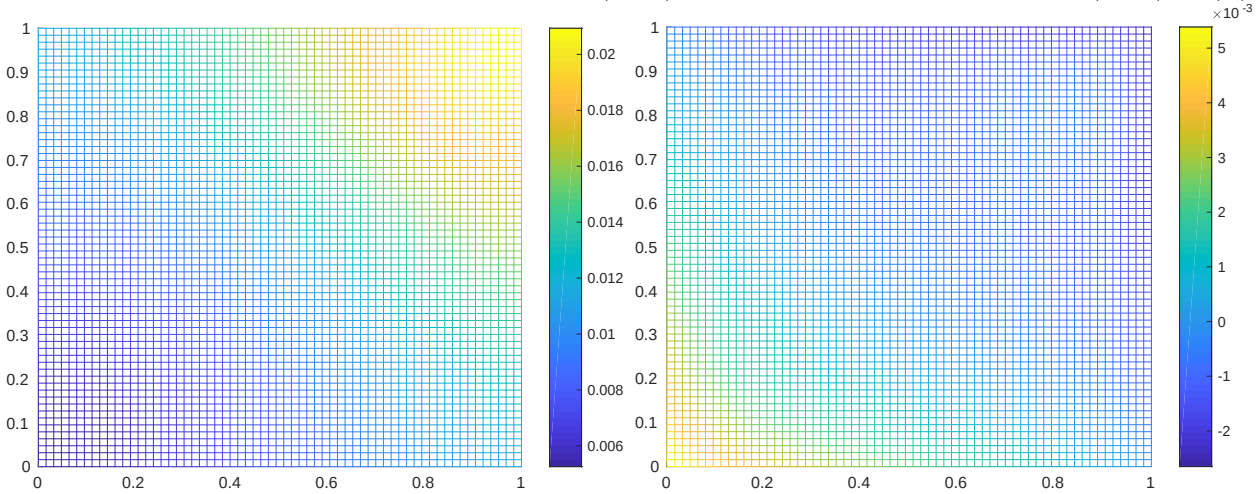
As expected, the complexity grows with the number of peaks, and for $m_0 \sim 20$ this approaches the estimated time of the full solver (should one have a sufficient amount of memory to run the latter). For a smaller number of peaks the TT solver is more efficient. Moreover, a small relative dispersion shows that it is quite insensitive to the particular realization of the initial distribution.

Figure 6: Left: initial cell density $z_0(\mathbf{x})$ for $m_0 = 10$. Right: control $u(\mathbf{x}, T/2)$.



The initial cell density for $m_0 = 10$ peaks and the transient control signal are shown in Fig. 6 (left and right, respectively), while the final density and the misfit are shown in Fig. 7. The unconstrained control takes negative values in the left bottom corner of the domain. However, this gives a more accurate fit of the cell density to the desired distribution than the

Figure 7: Left: cell density at the final time $z(\mathbf{x}, T)$. Right: misfit of the density, $z(\mathbf{x}, T) - \widehat{z}(\mathbf{x})$.



constrained control.

8 Concluding remarks

We have developed a preconditioned Gauss–Newton method for solving optimal control problems in chemotaxis, making use of an effective saddle point type preconditioner coupled with a suitable approximation of the pivoted Schur complement. This enables us to solve potentially huge-scale matrix systems, both without and with additional box constraints imposed on the control variable. Numerical results indicate considerable robustness with respect to the matrix dimension, as well as the parameters involved in the problem set-up.

Moreover, we have shown that the problem without box constraints is amenable to a faster solution using the low-rank tensor approximations of all vectors and matrices arising in the discretization. The nonlinearity of the problem can easily be tackled via cross approximation methods, provided that the functions are smooth. The low-rank decompositions are not very suitable for discontinuous functions, such as an indicator of an active set, arising in the problem of finding a constrained control or state. However, in the unconstrained case the low-rank algorithms are much faster and need much less memory than the straightforward solution of the Gauss–Newton equations. Depending on the "complexity" of the transient solution (and hence the tensor ranks), we can achieve a speedup of more than an order of magnitude.

The importance of the box constraints depends on the particular model. For example, if we can only control the inflow of the chemoattractant, it is reasonable to request a nonnegative control. However, if the laboratory setup allows one also to remove the chemoattractant, or to add a repellent, the negative control becomes physically realizable. This can provide a better control of the cell population, whereas the low-rank numerical algorithms allow a fast simulation of the required profile of the attractant/repellent, even on a low performance desktop.

Acknowledgements. SD and JWP gratefully acknowledge support from the Engineering and Physical Sciences Research Council (UK) Fellowships EP/M019004/1 and EP/M018857/2, respectively.

References

[1] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll. Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data. *Comput. Method. Appl. M.*, 304:26–54, 2016.

[2] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.

[3] S. Dolgov, J. W. Pearson, D. V. Savostyanov, and M. Stoll. Fast tensor product solvers for optimization problems with fractional differential equations as constraints. *Appl. Math. Comput.*, 273:604–623, 2016.

[4] S. Dolgov and M. Stoll. Low-rank solution to an optimization problem constrained by the Navier–Stokes equations. *SIAM J. Sci. Comput.*, 39(1):A255–A280, 2017.

[5] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov. Computation of extreme eigenvalues in higher dimensions using block tensor train format. *Computer Phys. Comm.*, 185(4):1207–1216, 2014.

[6] H. Egger, J.-F. Pietschmann, and M. Schlottbom. Identification of chemotaxis models with volume-filling. *SIAM J. Appl. Math.*, 75(2):275–288, 2015.

[7] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods, Part I. *Numer. Math.*, 3:147–156, 1961.

[8] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods, Part II. *Numer. Math.*, 3:157–168, 1961.

[9] W. Hackbusch. *Tensor Spaces And Numerical Tensor Calculus*. Springer-Verlag, Berlin, 2012.

[10] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012.

[11] I. C. F. Ipsen. A note on preconditioning nonsymmetric matrices. *SIAM J. Sci. Comput.*, 23(3):1050–1051, 2001.

[12] K. Ito and K. Kunisch. Semi-smooth Newton methods for state-constrained optimal control problems. *Syst. Control Lett.*, 50:221–228, 2003.

[13] V. A. Kazeev and B. N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM J. Matrix Anal. Appl.*, 33(3):742–758, 2012.

[14] V. A. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov. Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity. *SIAM J. Sci. Comput.*, 35(3):A1511–A1536, 2013.

[15] E. F. Keller and L. A. Segel. Model for chemotaxis. *J. Theor. Biol.*, 30(2):225–234, 1971.

[16] B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Approx.*, 34(2):257–280, 2011.

[17] B. N. Khoromskij. Tensor numerical methods for multidimensional PDEs: Theoretical analysis and initial applications. *ESAIM: Proc.*, 48:1–28, 2015.

[18] Y. A. Kuznetsov. Efficient iterative solvers for elliptic finite element problems on non-matching grids. *Russ. J. Numer. Anal. M.*, 10(3):187–212, 1995.

[19] D. Lebiedz and U. Brandt-Pollmann. Manipulation of self-aggregation patterns and waves in a reaction–diffusion system by optimal boundary control strategies. *Phys. Rev. Lett.*, 91(20):208301, 2003.

[20] A. Y. Mikhalev and I. V. Oseledets. Rectangular maximum-volume submatrices and their applications. *Linear Algebra Appl.*, 538:187–211, 2018.

[21] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.

[22] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[23] I. V. Oseledets and E. E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.

[24] J. W. Pearson and J. Gondzio. Fast interior point solution of quadratic programming problems arising from PDE-constrained optimization. *Numer. Math.*, 137(4):959–999, 2017.

[25] J. W. Pearson and M. Stoll. Fast iterative solution of reaction–diffusion control problems arising from chemical processes. *SIAM J. Sci. Comput.*, 35(5):B987–B1009, 2013.

[26] J. W. Pearson, M. Stoll, and A. J. Wathen. Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems. *SIAM J. Matrix Anal. Appl.*, 33(4):1126–1152, 2012.

[27] J. W. Pearson, M. Stoll, and A. J. Wathen. Preconditioners for state-constrained optimal control problems with Moreau–Yosida penalty function. *Numer. Lin. Alg. Appl.*, 21(1):81–97, 2014.

[28] J. W. Pearson and A. J. Wathen. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numer. Lin. Alg. Appl.*, 19(5):816–829, 2012.

[29] J. W. Pearson and A. J. Wathen. Fast iterative solvers for convection–diffusion control problems. *Electron. Trans. Numer. Anal*, 40:294–310, 2013.

[30] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Info. Comput.*, 7(5):401–430, 2007.

[31] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Universität Heidelberg, 2011.

[32] T. Rees, H. S. Dollar, and A. J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM J. Sci. Comput.*, 32(1):271–298, 2010.

[33] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[34] D. V. Savostyanov. Quasioptimality of maximum-volume cross interpolation of tensors. *Linear Algebra Appl.*, 458:217–244, 2014.

[35] D. V. Savostyanov and I. V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *Proceedings of 7th International Workshop on Multidimensional Systems (nDS)*. IEEE, 2011.

[36] R. Schneider and A. Uschmajew. Approximation rates for the hierarchical tensor format in periodic sobolev spaces. *J. Complexity*, 30(2):56–71, 2014.

[37] J. Schöberl and W. Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM J. Matrix Anal. Appl.*, 29(3):752–773, 2007.

[38] U. Schollwöck. The density-matrix renormalization group. *Rev. Mod. Phys.*, 77(1):259–315, 2005.

[39] M. Stoll, J. W. Pearson, and P. K. Maini. Fast solvers for optimal control problems from pattern formation. *J. Comput. Phys.*, 304:27–45, 2016.

[40] E. E. Tyrtyshnikov. Tensor approximations of matrices generated by asymptotically smooth functions. *Sb. Math.*, 194(6):941–954, 2003.

[41] A. J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA J. Numer. Anal.*, 7(4):449–457, 1987.

[42] A. J. Wathen and T. Rees. Chebyshev semi-iteration in preconditioning for problems including the mass matrix. *Electron. Trans. Numer. Anal.*, 34:125–135, 2009.

[43] W. Zulehner. Nonstandard norms and robust estimates for saddle point problems. *SIAM J. Matrix Anal. Appl.*, 32(2):536–560, 2011.